Computational Statistics Lab6

Group6: Araya Eamrurksiri and Oscar Pettersson

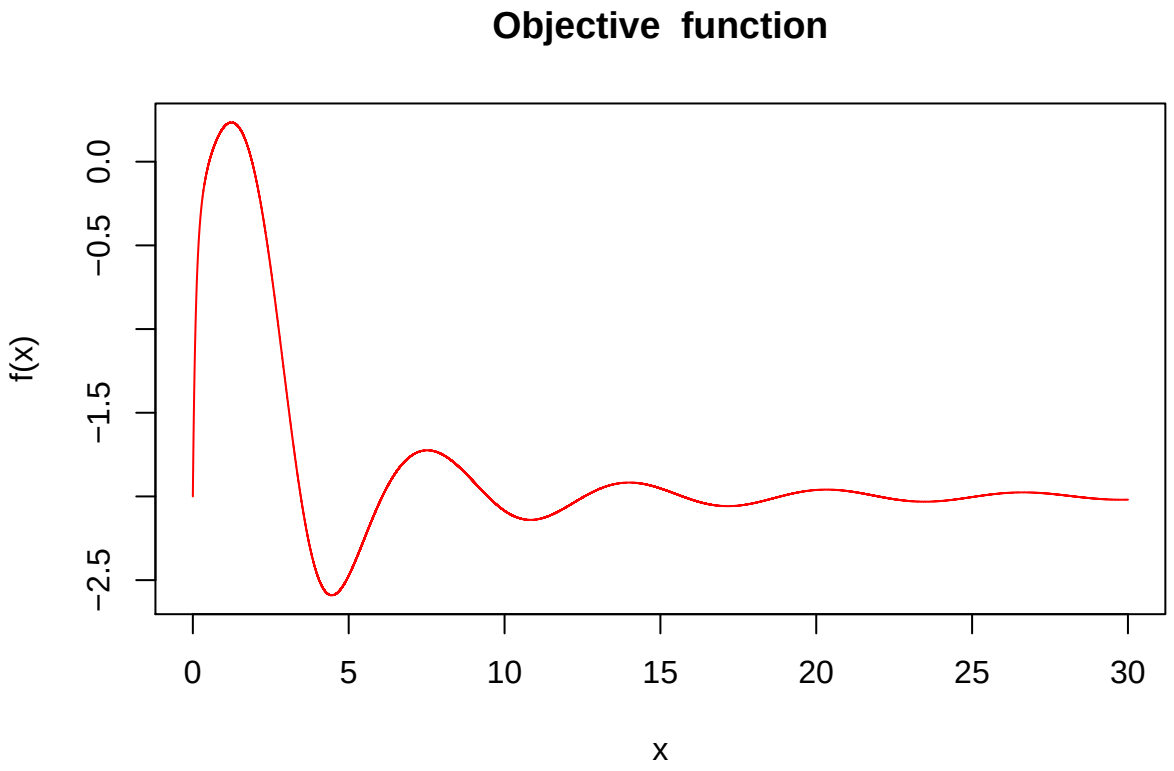
March 16, 2016

Assignment1: Genetic algorithm

in this assignment, you will try to perform one-dimensional maximization with the help of a genetic algorithm.

1.1 Define the function $f(x) = \frac{x^2}{e^x} - 2e^{\frac{-9sin(x)}{x^2+x+1}}$

We let the objective function $f(x) = \frac{x^2}{e^x} - 2e^{\frac{-9sin(x)}{x^2+x+1}}$. It has got a lot of local maxima but the global maximum for $x >= 0$ is about 0.234853 when $x \approx 1.2392$.



1.2 Define the function "crossover" that for two scalar x and y returns their "kid" as $(x + y)/2$

We have a function $crossover(x, y) = \frac{x+y}{2}$. So, the regular "child" is merely the average of its "parents".

1.3 Define the function "mutate" that for scalar x returns the result of the integer division x^2 *mod* 30. (Operation "mod" is denoted in R as "%%")

We also add a function $mutate(x) = (x^2)\ mod\ 30$. The child will be mutated with the probability given by the user.

1.4 Write a function that depends on the parameters *maxiter* and *mutprob* and:

a. Plots function f in the range from 0 to 30. Do you see any maximum value?

b. Defines an initial population for the genetic algorithm as $X = (0, 5, 10, 15, ..., 30)$

c. Computes vector "Values" that contains the function values for each population point.

d. Performs *maxiter* iterations where at each iteration

   i. Two indexes are randomly sampled from the current population, they are further used as parents (use *sample()*)

   ii. One index with the smallest objective function is selected from the current population, the point is referred to as victim (use *order()*)

   iii. Parents are used to produce a new kid by crossover. Mutate this kid with probability *mutprob*. (use *crossover(), mutate()*)

   iv. The victim is replaced by the kid in the population and the list "Values" is updated

   v. The current maximal value of the objective function is saved

e. Final observations are added to the current plot and marked by some other color.

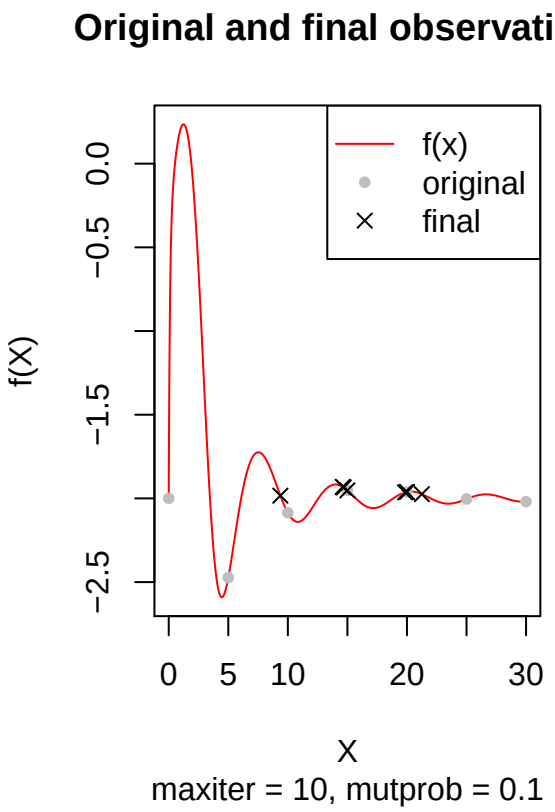
We implement the function *gen_algo* that runs the genetic algorithm by randomly sampling two parents from the existing population and mutating their child with the probability *mutprob*. The least fit observation will be replaced by this child. The process is iterated *maxiter* times. As a measure of fitness, it uses the objective function f from 1.1 and the initial population is $X = \{0, 5, ..., 30\}$.

```
gen_algo <- function(maxiter, mutprob) {

  # a - Plot
  X <- seq(from = 0, to = 30, by = 0.01)
  plot(f(X) ~ X, type = "l", col = "red", main = "Original and final observations",
       sub = paste("maxiter = ", maxiter, ", mutprob = ", mutprob, sep = ""))

  # b - Initial population
  X <- seq(from = 0, to = 30, by = 5)
  points(f(X) ~ X, col = "gray", pch = 20)
  # c - Initial values of objective function
  fX <- f(X)

  # d - Iterate
  max_f <- rep(NA_real_, maxiter)
  for (i in 1:maxiter) {

    # i - Sample two "parents"
    parents <- sample(x = X, size = 2, replace = TRUE)

    # ii - Find index of "victim"
```

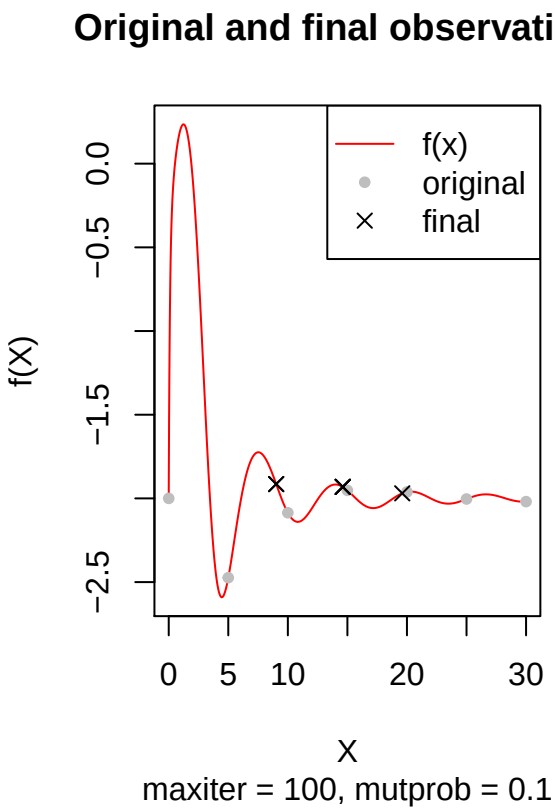
```
    victim_index <- which.min(fX)

    # iii - Get child, possibly mutated
    new_child <- crossover(x = parents[1], y = parents[2])
    if (runif(1) < mutprob) new_child <- mutate(new_child)

    # iv - Replace "victim" with new child
    X[victim_index] <- new_child
    fX[victim_index] <- f(X[victim_index])

    # v - Calculate a current max
    max_f[i] <- max(fX)
  }

  # e - Add final observations
  points(fX ~ X, pch = 4)
  legend(x = "topright", legend = c("f(x)", "original", "final"), pch = c(0,20,4),
         col = c("red", "gray", "black"), lty = c(1,0,0), pt.cex = c(0, 1, 1))
  return(list(final_pop = X, final_fitness = fX, max_fXs = max_f))
}
```

1.5 Run your code with different combinations of *maxiter=10, 100* and *mutprob=0.1,0.5,0.9*. Observe the initial population and final population. Conclusions?

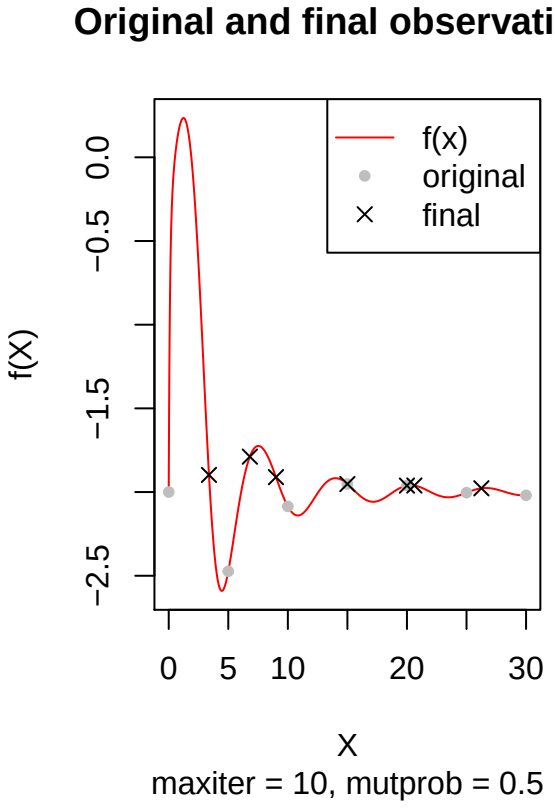
The function is run with the same seed for all combinations of $maxiter \in \{10, 100\}$ and $mutprob \in \{0.1, 0.5, 0.9\}$:



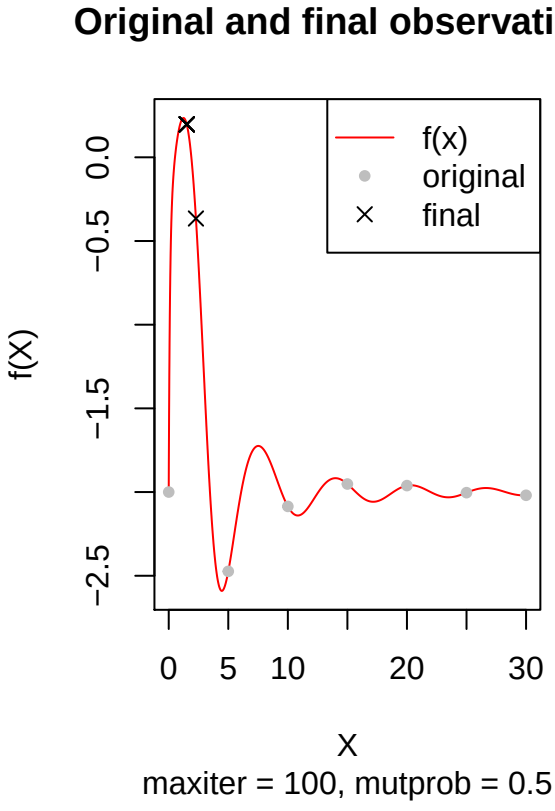
maxiter = 10, mutprob = 0.1



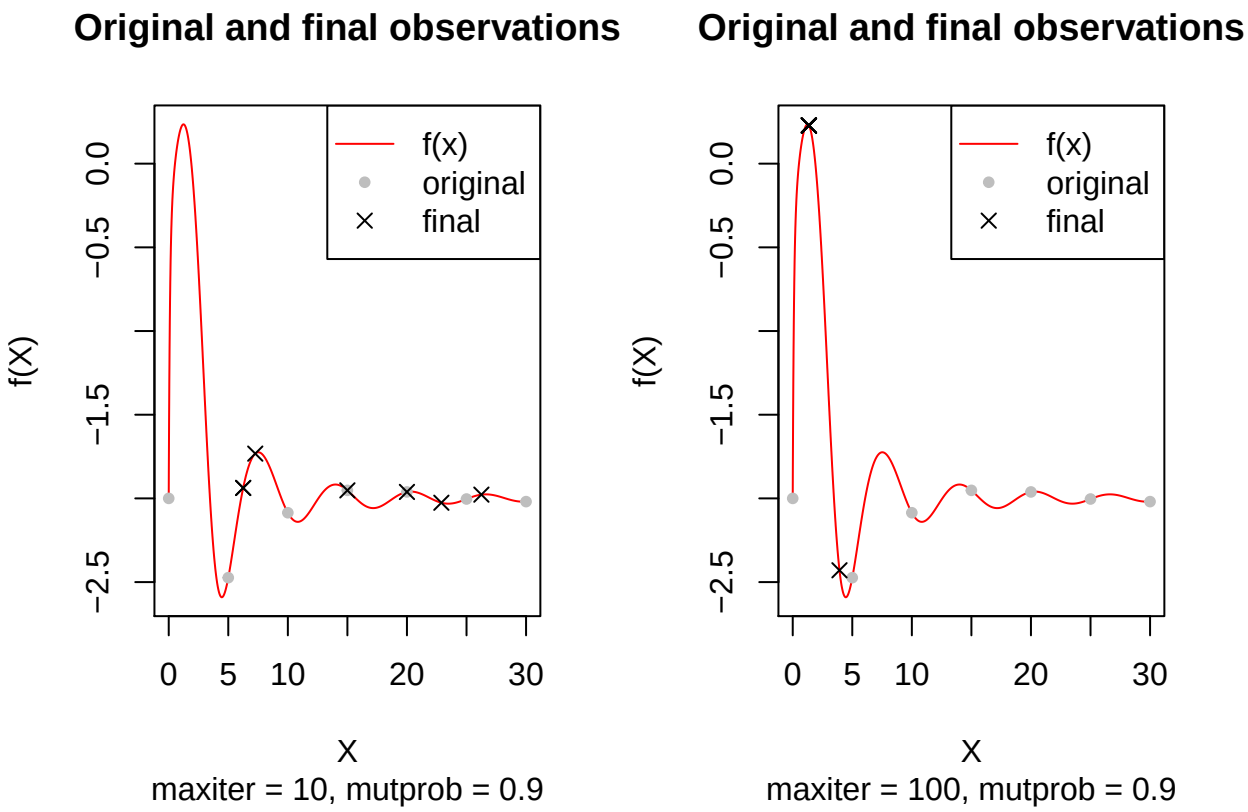
maxiter = 100, mutprob = 0.1



maxiter = 10, mutprob = 0.5

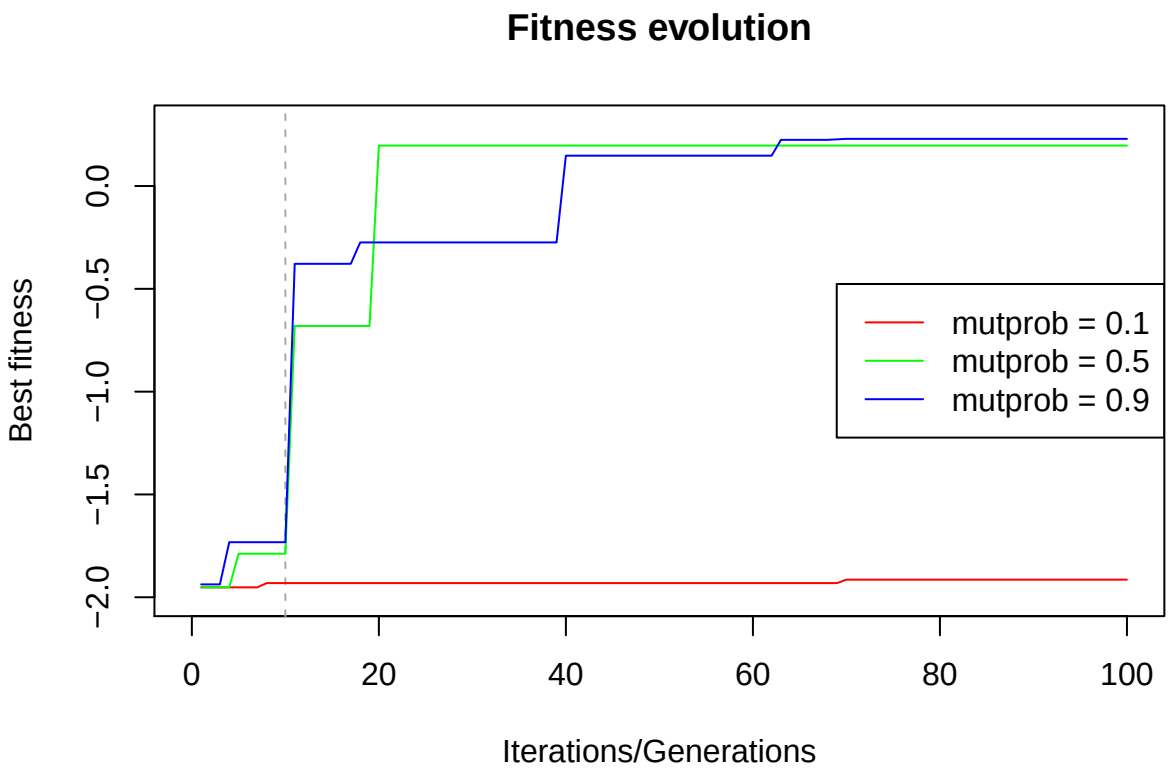


maxiter = 100, mutprob = 0.5




```
##   maxiter mutprob   obs1   obs2   obs3   obs4   obs5   obs6   obs7
## 1      10     0.1 14.688 21.250  9.375 15.000 20.000 14.609 19.844
## 2      10     0.5  9.004  3.379 26.250 15.000 20.000  6.811 20.625
## 3      10     0.9 22.891  6.250 26.250 15.000 20.000  6.250  7.265
## 4     100     0.1  9.022  9.022 19.606 14.609 14.609 14.609 14.609
## 5     100     0.5  2.283  1.511  1.511  1.511  1.511  1.511  1.511
## 6     100     0.9  1.364  3.925  1.343  1.362  1.343  1.364  1.364
```

When *maxiter* is equal to 10, some values of the final population do not change their values very much. Other observations also have quite similar value as the initial population. Mostly, we don't get close to the global optimum when using *maxiter=10*. On the other hand, when *maxiter=100*, every observation in the final population has changed its value and is much differ from the initial population. The X-values of each final population is rather low and the algorithm is able to find a global optimum or rather a close one.

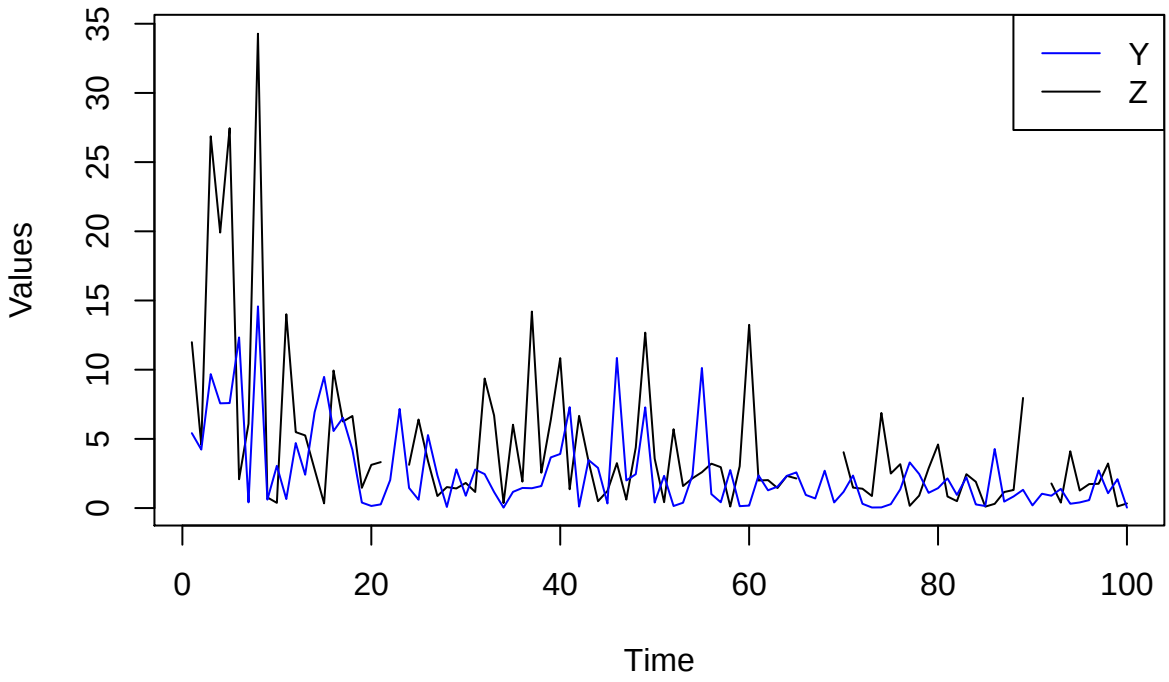


We see that running the algorithm for 100 iterations yields much better results than when only 10 iterations are run (since all runs had the same seed). But when *mutprob* is low, not even 100 iterations give a good final result. Running the algorithm for an even higher number of iterations seems to give even better results, though this is not presented here.

Assignment2: EM algorithm

Data file physical.csv describes a behavior of two related physical processes $Y = Y(X)$ and $Z = Z(X)$

2.1 Make a time series plot describing dependence of Z and Y versus X. Does it seem that two processes are related to each other? What can you say about the variation of the response values with respect to X?



It seems that these two process are somewhat related to each other. The variation of the responses are quite varied over the period of time. They have high value in the beginning and gradually decrease with some fluctuation. Moreover, it can be seen that there are some missing values in Z.

2.2 Note that there are some missing values of Z in the data which implies problems in estimating models by maximum likelihood. Use the following model

$$Y_i \sim Exp\left(\frac{X_i}{\lambda}\right), Z_i \sim Exp\left(\frac{X_i}{2\lambda}\right)$$

where λ is some unknown parameter to derive an EM algorithm that estimates λ.

Likelihood:

$P(Y_i|\lambda) = \frac{\prod_{i=1}^{n} X_i}{\lambda^n} exp(-\frac{1}{\lambda}\sum_{i=1}^{n} Y_i X_i)$

$P(Z_i|\lambda) = \frac{\prod_{i=1}^{n} X_i}{(2\lambda)^n} exp(-\frac{1}{2\lambda}\sum_{i=1}^{n} Z_i X_i)$

$$P(Y_i, Z_i|\lambda) = P(Y_i|\lambda) \cdot P(Z_i|\lambda) = \frac{(\prod_{i=1}^{n} X_i)^2}{(2\lambda^2)^n} exp\left(-\frac{\sum_{i=1}^{n} Y_i X_i}{\lambda} - \frac{\sum_{i=1}^{n} Z_i X_i}{2\lambda}\right)$$

As Z contains some missing values, we have to divide it into two terms, observed and unobserved term.
$Z = \{Z_i; i \in o\}$ and $Z = \{Z_j; j \in m\}$
Therefore,

$$P(Y_i, Z_i|\lambda) = \frac{(\prod_{i=1}^{n} X_i)^2}{(2\lambda^2)^n} exp\left(-\frac{\sum_{i=1}^{n} Y_i X_i}{\lambda} - \frac{\sum_{i\in o} Z_i X_i}{2\lambda} - \frac{\sum_{j\in m} Z_j X_j}{2\lambda}\right)$$

Log-likelihood:

$$logP(Y_i, Z_i|\lambda) = 2ln\left(\prod_{i=1}^{n} X_i\right) - nln(2) - 2ln(\lambda) - \frac{\sum_{i=1}^{n} Y_i X_i}{\lambda} - \frac{\sum_{i\in o} Z_i X_i}{2\lambda} - \frac{\sum_{j\in m} Z_j X_j}{2\lambda}$$

Expectation:

$E_{Z|\lambda^t,Y} logP = 2ln(\prod_{i=1}^{n} X_i) - nln(2) - 2ln(\lambda) - \frac{\sum_{i=1}^{n} Y_i X_i}{\lambda} - \frac{\sum_{i\in o} Z_i X_i}{2\lambda} - \frac{\sum_{j\in m} X_j E(Z_j|\lambda^t)}{2\lambda}$

As $E(Z_j|\lambda^t) = \frac{2\lambda^t}{X_j}$, we got $\frac{\sum_{j\in m} X_j E(Z_j|\lambda^t)}{2\lambda} = \frac{\sum_{j\in m} X_j(\frac{2\lambda^t}{X_j})}{2\lambda}$
Therefore,

$$E_{Z|\lambda^t,Y} logP = 2ln\left(\prod_{i=1}^{n} X_i\right) - nln(2) - 2ln(\lambda) - \frac{\sum_{i=1}^{n} Y_i X_i}{\lambda} - \frac{\sum_{i\in o} Z_i X_i}{2\lambda} - \frac{\lambda^t|m|}{\lambda};$$

$|m|$ = *amount of unobserved term*

Maximization:

Then, taking the partial derivative with respect to λ

$$\frac{\partial E logP}{\partial \lambda} = -\frac{2n}{\lambda} + \frac{\sum_{i=1}^{n} Y_i X_i}{\lambda^2} + \frac{\sum_{i\in o} Z_i X_i}{2\lambda^2} + \frac{\lambda^t|m|}{\lambda^2}$$

setting the partial derivative to zero, we got

$$\lambda^{t+1} = \frac{1}{2n}\left(\sum_{i=1}^{n} Y_i X_i + \frac{\sum_{i\in o} Z_i X_i}{2} + \lambda^t|m|\right)$$

2.3 Implement this algorithm in R and use $\lambda_0 = 100$ and convergence criterion "stop if the change in λ is less than 0.001". What is the optimal λ and how many iterations were required to compute it?

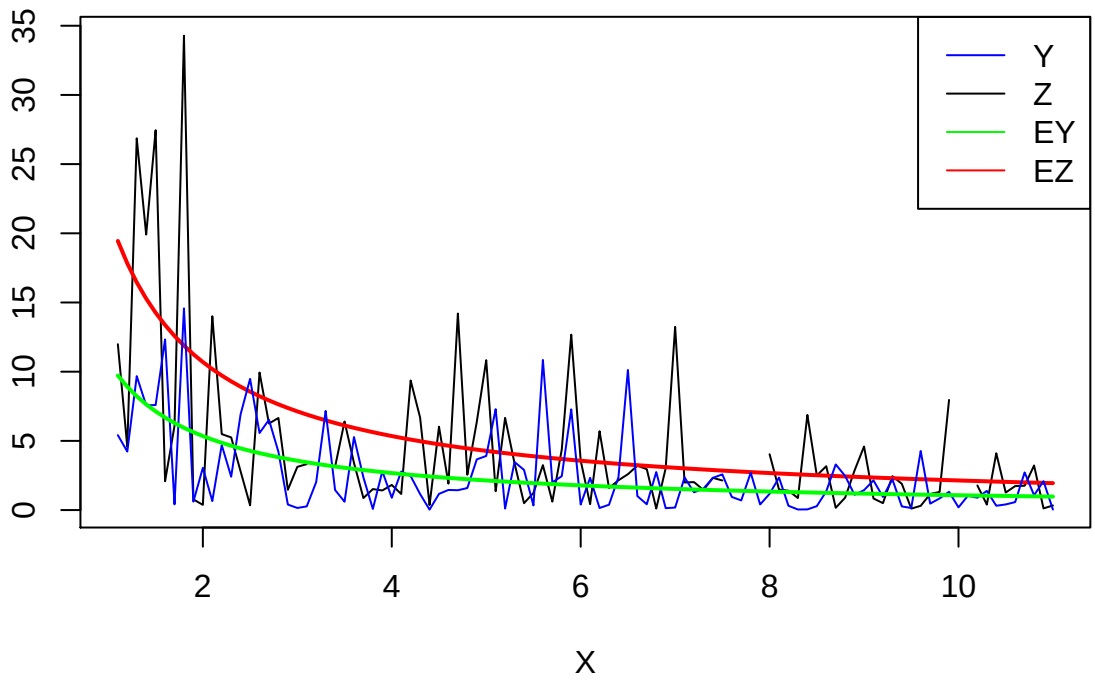
In this step, the EM algorithm is being implemented as follows:

```
EM <- function(X,Y,Z){
    Xobs <- X[!is.na(Z)]
    Xmiss <- X[is.na(Z)]
    Zobs <- Z[!is.na(Z)]
    n <- length(X)
    m <- length(Xmiss)

    #Initial values
    lambdat <- 100

    #Define log-likelihood function
    ll <- function(X, Y, Zobs, Xobs, lambda, n){
        2*sum(log(X)) - ( n*log(2) ) - ( 2*n*log(lambda) ) -
            ( sum(Y*X)/lambda ) - ( sum(Zobs*Xobs)/(2*lambda) )
    }

    repeat{
        #M-step
        lambdat1 <- ( sum(Y*X) + (sum(Zobs*Xobs)/2) + (lambdat*m) )/(2*n)

        #Compute log-likelihood using current estmates
        llt <- ll(X, Y, Zobs, Xobs, lambdat1, n)

        #Print current parameter values and likelihood
        cat("lambda =",lambdat1, "log-likelihood =", llt, "\n")

        #Stop if converged
        if(abs(lambdat1 - lambdat) < 0.001) break
        lambdat <- lambdat1
    }
}

EM(physical$X, physical$Y, physical$Z)
```

```
## lambda = 14.26782 log-likelihood = -414.9261
## lambda = 10.83853 log-likelihood = -405.4854
## lambda = 10.70136 log-likelihood = -405.3667
## lambda = 10.69587 log-likelihood = -405.3625
## lambda = 10.69566 log-likelihood = -405.3624
```

The estimated optimal $\hat{\lambda}$ is equal to 10.69566 and 5 iterations are required to find this value and the log-likelihood does indeed get better as $\hat{\lambda}$ gets updated.

2.4 Plot EY and EZ versus X in the same plot as Y and Z versus X and comment whether the computed λ seems to be reasonable.



The computed lambda we have got in the previous step seems to be reasonable as it fit with the original data, both in Y and Z, quite well.

Appendix

Contribution

For the first assignment, the code and most of the text come from Oscar's report and for the second, the code and most of the text is from Araya.

R-code

```
f <- function(x) (x ^ 2 / exp(x)) - 2 * exp((-9 * sin(x)) / (x ^ 2 + x + 1))
x <- seq(from = 0, to = 30, by = 0.0001)
y <- f(x)
plot(f(x) ~ x, type = "l", col = "red", main = "Objective  function")
# Max for x = 1.239156|25
crossover <- function(x, y) (x + y) / 2
mutate <- function(x) (x ^ 2) %% 30
gen_algo <- function(maxiter, mutprob) {

  # a - Plot
  X <- seq(from = 0, to = 30, by = 0.01)
  plot(f(X) ~ X, type = "l", col = "red", main = "Original and final observations",
       sub = paste("maxiter = ", maxiter, ", mutprob = ", mutprob, sep = ""))

  # b - Initial population
  X <- seq(from = 0, to = 30, by = 5)
  points(f(X) ~ X, col = "gray", pch = 20)
  # c - Initial values of objective function
  fX <- f(X)

  # d - Iterate
  max_f <- rep(NA_real_, maxiter)
  for (i in 1:maxiter) {

    # i - Sample two "parents"
    parents <- sample(x = X, size = 2, replace = TRUE)

    # ii - Find index of "victim"
    victim_index <- which.min(fX)

    # iii - Get child, possibly mutated
    new_child <- crossover(x = parents[1], y = parents[2])
    if (runif(1) < mutprob) new_child <- mutate(new_child)

    # iv - Replace "victim" with new child
    X[victim_index] <- new_child
    fX[victim_index] <- f(X[victim_index])

    # v - Calculate a current max
    max_f[i] <- max(fX)
  }

  # e - Add final observations
```

```
  points(fX ~ X, pch = 4)
  legend(x = "topright", legend = c("f(x)", "original", "final"), pch = c(0,20,4),
         col = c("red", "gray", "black"), lty = c(1,0,0), pt.cex = c(0, 1, 1))
  return(list(final_pop = X, final_fitness = fX, max_fXs = max_f))
}
par(mfrow = c(1, 2))
set.seed(12345)
m10_p.1 <- gen_algo(maxiter = 10, mutprob = 0.1)
set.seed(12345)
m100_p.1 <- gen_algo(maxiter = 100, mutprob = 0.1)
set.seed(12345)
m10_p.5 <- gen_algo(maxiter = 10, mutprob = 0.5)
set.seed(12345)
m100_p.5 <- gen_algo(maxiter = 100, mutprob = 0.5)
set.seed(12345)
m10_p.9 <- gen_algo(maxiter = 10, mutprob = 0.9)
set.seed(12345)
m100_p.9 <- gen_algo(maxiter = 100, mutprob = 0.9)
par(mfrow = c(1, 1))
dat <- t(data.frame(round(m10_p.1$final_pop,3), round(m10_p.5$final_pop,3), round(m10_p.9$final_pop,3),
rownames(dat) <- c("1","2","3","4","5","6")
colnames(dat) <- c("obs1","obs2","obs3","obs4","obs5","obs6","obs7")
dat <- cbind(maxiter=c(rep(10,3),rep(100,3)),mutprob=c(rep(c(0.1,0.5,0.9),2 )), dat)
print(dat)
plot(NULL, xlim = c(0, 100), ylim = c(-2, .3), xlab = "Iterations/Generations", ylab = "Best fitness", m
abline(v = 10, col = "darkgray", lty = "dashed")
points(m100_p.1$max_fXs ~ c(1:100), type = "l", col = "red")
points(m100_p.5$max_fXs ~ c(1:100), type = "l", col = "green")
points(m100_p.9$max_fXs ~ c(1:100), type = "l", col = "blue")
legend("right",, c("mutprob = 0.1", "mutprob = 0.5", "mutprob = 0.9"), col = c("red", "green", "blue"),
physical <- read.csv("physical.csv")

#1
plot(ts(physical$Z), ylab="Values")
lines(ts(physical$Y), col="blue")
legend("topright", c("Y","Z"), col=c("blue","black"), lwd=1)
EM <- function(X,Y,Z){
    Xobs <- X[!is.na(Z)]
    Xmiss <- X[is.na(Z)]
    Zobs <- Z[!is.na(Z)]
    n <- length(X)
    m <- length(Xmiss)

    #Initial values
    lambdat <- 100

    #Define log-likelihood function
    ll <- function(X, Y, Zobs, Xobs, lambda, n){
        2*sum(log(X)) - ( n*log(2) ) - ( 2*n*log(lambda) ) -
            ( sum(Y*X)/lambda ) - ( sum(Zobs*Xobs)/(2*lambda) )
    }

    repeat{
```

```
        #M-step
        lambdat1 <- ( sum(Y*X) + (sum(Zobs*Xobs)/2) + (lambdat*m) )/(2*n)

        #Compute log-likelihood using current estmates
        llt <- ll(X, Y, Zobs, Xobs, lambdat1, n)

        #Print current parameter values and likelihood
        cat("lambda =",lambdat1, "log-likelihood =", llt, "\n")

        #Stop if converged
        if(abs(lambdat1 - lambdat) < 0.001) break
        lambdat <- lambdat1
    }
}

EM(physical$X, physical$Y, physical$Z)
#optimal lambda 10.69566
lambda <- 10.69566
X <- physical$X
Y <- physical$Y
Z <- physical$Z

EY <- lambda/X
EZ <- (2*lambda)/X

plot(X, Z, type="l", ylab="")
lines(X, EZ, col="red", lwd=2)
lines(X, Y, col="blue")
lines(X, EY, col="green", lwd=2)
legend("topright", c("Y","Z","EY","EZ"), col=c("blue","black","green","red"), lwd=1)
## NA
```